DOI: https://doi.org/10.15276/aait.07.2024.12
UDC 004.056

The improvement of web-application SDL process to prevent Insecure Design vulnerabilities

Oleksandr A. Revniuk[1)]
ORCID: https://orcid.org/0009-0005-0511-5354; revo0708@gmail.com
Nataliya V. Zagorodna[1)]
ORCID: https://orcid.org/0000-0002-1808-835X; zagorodna_n@tntu.edu.ua. Scopus Author ID: 57189380553
Ruslan O. Kozak[1)]
ORCID: https://orcid.org/0000-0003-1323-0801; ruslank@tntu.edu.ua. Scopus Author ID: 57193443499
Mikolaj P. Karpinski[2)]
ORCID: https://orcid.org/0000-0002-8846-332X; mikolaj.karpinski@up.krakow.pl. Scopus Author ID: 57202467671
Liubomyr O. Flud[3)]
ORCID: https://orcid.org/0000-0002-8347-4265; flud@nltu.edu.ua. Scopus Author ID: 57202467671
[1)] Ternopil Ivan Puluj National Technical University, 56, Ruska Str. Ternopil, 46001, Ukraine
[2)] University of the National Education Commission, 2, Podchorążych Str. Krakow, 30-084, Poland
[3)] Ukrainian National Forestry University, 103, Gen. Chuprynky Str. Lviv, 79057, Ukraine

ABSTRACT

According to the latest “OWASP Top Ten” list, “Insecure Design” vulnerability is one of the key factors affecting the level of data protection and functional reliability. Heightening attention to this issue is pertinent as this vulnerability is appeared to be the first time in OWASP list and just briefly described there. This study aims to identify and analyze the architectural vulnerabilities of web applications arising from “Insecure Design”. The goal is not only to identify specific vulnerabilities in the web applications design and implementation process but also to develop a detailed list of recommendations, that will help not only avoid similar problems in the future but to create a good background for safe web applications development from the start point. In order to construct a systematic approach to security at all stages of development, recommendations from the Software Development Life Cycle standard are considered here. Special attention is given to integrating security principles at all stages of the development lifecycle. The analysis is based on examining existing architectural solutions, studying vulnerabilities, and developing methods for their mitigation. The developed set of recommendations to enhance the security of web applications includes measures for architectural design, verification and validation processes, and early detection of potential vulnerabilities. Significant attention is paid to developing secure code, implementing security policies, and organizing training for developers. The research emphasizes the importance of integrating security into the web application development process from the beginning. The scientific novelty lies in the systematization and development of approaches to detect and mitigate architectural vulnerabilities caused by “Insecure Design”. The practical significance of the paper is expressed in enhancing the security level of web applications, reducing risks for businesses and users, and fostering a culture of security among developers.

Keywords: Insecure design; web applications; secure development lifecycle; security practices; application vulnerability; multi-layered structure

For citation: **Revniuk O., Zagorodna N., Kozak R., Karpinski M., Flud L. “The improvement of web-application SDL process to prevent Insecure Design vulnerabilities”. *Applied Aspects of Information Technology.* 2024; Vol. 7 No. 2: 162–174. DOI: https://doi.org/10.15276/aait.07.2024.12**

INTRODUCTION

In the era of globalization and digital transformation, web applications have evolved into an indispensable component of various aspects of people's daily lives. From personal blogs to complex corporate systems, their application penetrates every segment of our activities, causing a significant increase in their popularity. Such growth not only confirms their importance in the modern digital environment but also emphasizes the significant excess of the number of web applications over traditional applications. Consequently, the increase in their usage makes web applications attractive targets for cyber attacks and fraudulent actions, which in turn leads to an increase in incidents related to the leakage of personal data of users and vulnerabilities of security systems.

In this context, companies engaged in the development of web applications must pay special attention to aspects of security and data confidentiality. The application of advanced practices and innovative technologies in the field of

cyber security becomes a necessity to ensure protection against a wide range of threats. This includes the implementation of comprehensive authentication mechanisms, data encryption, regular software updates, as well as the development and implementation of incident response strategies. Regular cases of security breaches demonstrate the critical need for constant analysis, evaluation, and improvement of existing protective measures.

1. ANALYSIS OF LITERARY DATA AND PROBLEM STATEMENT

Successful attacks on web applications, which are increasing annually, result in financial losses for companies and damage to their reputation. Most often, users become the victims of cyberattacks aimed at seizing computational resources, accounts, or user data. Users typically trust web applications from well-known brands or institutions more, believing that large corporations devote adequate attention to the security of their applications. However, large companies are also victims of cyberattacks [1]. News reports about attacks on such corporations often dominate the headlines [2].

Fig. 1 shows the 10 biggest data breaches of recent years. However, web applications on a smaller scale frequently become victims too. According to SiteLock, which analyzed 7 million websites, modern websites undergo an average of 94 cyberattacks per day and receive about 2,608 bot visits per week [3].

Attacks that are directed at a specific application and continue until a certain result is achieved are called “Targeted attacks”. Typically, they choose one or a few victims and use the maximum amount of resources to find vulnerabilities and successfully hack. Often, the victims are websites that come into the attacker's view accidentally, as a result of using automated tools for conducting attacks. Attackers exploit vulnerabilities at the hardware, software, and communication levels.

The classification of modern vulnerabilities is undertaken by the OWASP community (Open Web Application Security Project). This is an international non-profit organization that specializes in analyzing and improving the security of software, whose members identify the most dangerous vulnerabilities based on statistics and research, allowing them to focus on the most critical aspects of software security. The research of the OWASP community is an important source of information for developers and software security professionals, as it enables the identification and mitigation of potential threats to the security of web applications at early stages of their development.

In 2003, OWASP first proposed a list of the most dangerous web application vulnerabilities, called the “OWASP Top Ten”. All products of this community are free and open source. Thousands of companies and professionals worldwide collaborate with it, using it as a standard reference document for testers and web application developers. The research of the OWASP community is an important source of information for developers and software security professionals, as it enables the identification and mitigation of potential threats to the security of web applications at early stages of their development.

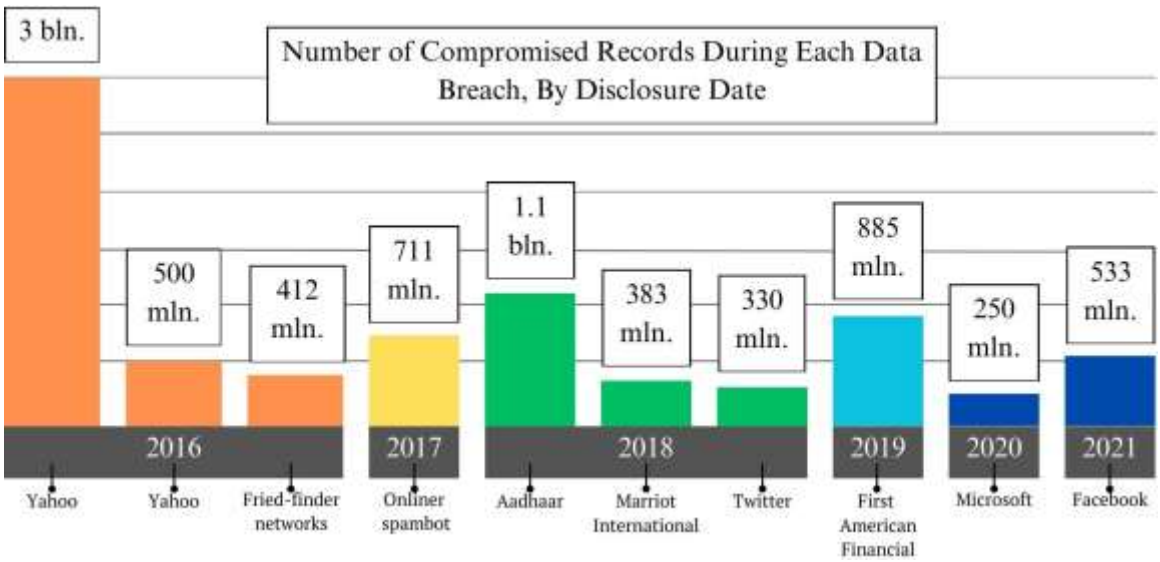

Fig. 1. **Top 10 largest data breaches in history**
Source: compiled by the [2]

The list of vulnerabilities is reviewed annually, but in practice, updates occur every three to four years. If updates happen more frequently, it indicates that attacks on web applications are advancing to a new level, becoming more prevalent, and necessitating a corresponding response from communities worldwide. The vulnerabilities in this list are ranked by frequency of occurrence, so their positions in the list change, also due to the emergence of new threats.

The latest update to the list of priority web application vulnerabilities occurred in 2021. From Fig. 2, it is evident that some vulnerability were transformed and appeared under new names, but new vulnerabilities also emerged [4, 5], [6].

Fig. 2. **Updated list of threats in 2021**
Source: **compiled by the authors**

In recent years, there has been an increase in incidents related to the "Insecure Design" vulnerability. This vulnerability occupies the fourth place in the list, indicating its significant impact on the security of web applications. A brief overview of data breaches provided below allows understanding the threat landscape and methods used by attackers to carry out attacks.

"Data of More Than 200 Million Twitter Users Is Leaked" – the data leak occurred in 2021 through the exploitation of a vulnerability in Twitter's API, which allowed users to enter email addresses and phone numbers to check if they were linked to a Twitter identifier. In this data leak, attackers combined publicly available data with private email addresses and phone numbers to create user profiles on Twitter using another social network API, which enabled obtaining identifiers from the company's publicly available data. They then matched the data to determine if a phone number or email address corresponded to the identifier. Despite the company fixing this vulnerability in January 2022, several threat actors have recently started distributing the data collections they obtained over a year ago for free. Thus, due to the large number of different APIs, it was possible to obtain unique user identifiers [7].

"Linux Malware Targets 30+ WordPress" "Plugins" is a Linux backdoor malware. It was discovered, capable of exploiting around 30 WordPress plugins to inject malicious JavaScript code and redirect users to harmful, malicious, and phishing websites created by attackers. This indicates that the imprudent use of outdated plugins with third-party or redundant functionality led to a serious vulnerability in the entire project [8].

"United States Office of Personnel Management – The most Flagrant" – On May 7, 2014, a Chinese group named X2 utilized stolen OPM credentials from Key Point to install malware for backdoor access left by the development team. As a result, there was a data breach containing information of a governmental nature.

Despite the large number of recorded incidents, it is important to note that the "Insecure Design" vulnerability is poorly formalized, and existing recommendations for its mitigation often are of a general nature and do not provide specific actions at different stages of web application development.

GOAL AND RESEARCH OBJECTIVES

The purpose of the research. The aim of the study is to enhance the security of web application by analysis and elimination of architectural vulnerabilities arising from the threat category known as "Insecure Design". The main idea is to identify specific weaknesses in the web application design and implementation process and suggest a list of mitigations, which could be embedded during

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

different stages of web application development lifecycle in order to secure the final software product.

The research tasks include: a detailed analysis of the vulnerabilities of "Insecure Design" will be carried out to understand the essence of the problem and its impact on the security of web applications. Based on conducted analysis, the existing approaches to overcoming this vulnerability, will be investigated and expanded in order to identify the most effective protection strategies. Using the research findings, expanded recommendations will be provided within the framework of the Secure Development Lifecycle (SDL) concept, which will facilitate the adaptation of security measures to various stages of web application development. As a result, specific approach on protecting against the "Insecure Design" vulnerability at the software development stage (SDL) will be formulated.

The object of research is the secure development lifecycle of web applications.

The subject of research are measures in the form of additions to the basic SDL diagram, to facilitate their practical implementation and ensure their systematic integration into web application development.

2. MAIN RESEARCH RESULTS

2.1. Web Application Structure

Planning the structure of web applications determines how their operation will be organized, which components will be used, how they will interact with each other and with other systems. To protect against threats, it is necessary to meticulously plan the structure of web applications, which will allow the identification and elimination of potential vulnerabilities associated with "Insecure Design" and apply best security practices at the architectural level.

Modern web applications are complex systems that can include dynamic content, interactive elements, databases, as well as integration with various services and applications [9, 10], [11, 12]. Web applications have a multi-layered structure, allowing them to be more flexible and functional. They can consist of various sections, subpages, modules, APIs, and other technologies that extend their capabilities and provide users with a wider range of services. Such complexity in web applications is reflected in their architecture and navigation, making them more adaptive and interactive for users. The generalized structure of a web application is illustrated in Fig. 3.

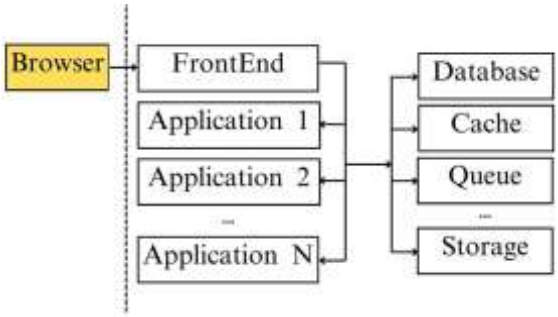

Fig. 3. **Structure of a modern web application**
Source: **compiled by the authors**

Communication between the client and web applications typically begins through a web browser, which interacts with the frontend part of the system. The frontend, in turn, interacts with backend applications that utilize various technologies for processing, storing, and providing the necessary information to the client [13, 14], [15].

Web applications comprise a set of components that work simultaneously and harmoniously. The client-side represents the output of developers' work, with which users directly interact in the browser. Thus, interaction with server components occurs through the HTTP protocol. Web applications use various protocols for interaction with different components. For example, the interaction of the "BackEnd" service with the "PostgreSQL" database may utilize a "native" protocol based on messages, supported at the TCP/IP level [16].

The server-side of modern web applications performs tasks responsible for managing the application's database and ensuring a continuous flow of information between the client interface and the server. The reliability and efficiency of the backend are crucial for providing a reliable user experience. Moreover, the architecture of the server-side often includes aspects such as scalability, security, and performance optimization to meet the demands of web applications.

However, it can be definitively stated that the complexity and functionality of web applications are increasing, and therefore, the number of potential vulnerabilities that could be exploited by malicious actors is also increasing.

Due to next-generation attacks and concealment methods, traditional defense systems, such as firewalls, intrusion detection systems, antivirus software, access control lists, etc., are not always effective. Often, the very fact of being unaware of a web application's compromise marks the beginning of problems related to the leakage of confidential information onto various "dark" markets, and later into the publicly accessible space of the Internet.

Therefore, when developing web applications using modern technologies and programming languages, it is important to remember that cybercriminals possess deep knowledge of operating systems, can quickly write malicious computer programs, and in a short time identify vulnerabilities in the latest web applications.

Attacks by malicious actors are typically directed at both components of web services: the client and server sides. However, attacks on the server side are particularly dangerous because the client may not even suspect the web application's compromise when entering a login and password, believing themselves to be secure. It is crucial, at the stage of creating a system or web application, to care for the security of each component of web applications.

2.2. Web Application Vulnerability Analysis

The security of the server-side is always a priority, given that the majority of vulnerabilities identified in the OWASP list are inherently related to the server side. This is a critical area of concern because malicious access to any internal component of a web service not only compromises that specific element but can also serve as a gateway for further attacks on the entire system. Such vulnerabilities underscore the paramount importance of rigorously focusing on the server side's security in web applications. The types of vulnerabilities affecting the server side can be broadly categorized into three main groups: logical vulnerabilities, which stem from flawed business logic or inadequate security measures; exploitative vulnerabilities, which can be abused through specific attack vectors like SQL injection or Cross-Site Scripting (XSS); and implementation error vulnerabilities, which arise from mistakes made during the development and deployment phases, such as misconfigurations or improper session management..

Logical vulnerabilities frequently emerge as a consequence of inadequate segregation of data access rights, the incorporation of excessive or potentially hazardous functionalities within the web application, and deviations from the intended logic of the project [17, 18], [19, 20]. Such vulnerabilities are particularly insidious because they exploit the inherent logic of the application rather than targeting lower-level coding mistakes or exploiting hardware weaknesses. Excessive or dangerous functionality refers to features that were not planned in the project requirements. For example, a developer may add console functionality to facilitate testing during development, and this functionality may remain in the product's "production" version upon deployment. Since this functionality could be known to malicious actors, they might use it to bypass the web application's protection rules. Logical vulnerabilities are often related to the application's business logic and are difficult to detect by someone not directly working with the service. Such issues should be identified by professionals during the product testing phase.

The vulnerability Insecure Direct Object Reference (IDOR) is quite common due to the simplicity of its implementation [21]. According to research [22, 23], [24], it is particularly prevalent in web applications with APIs – developers often encounter this vulnerability when protecting client software, as illustrated in Fig. 4.

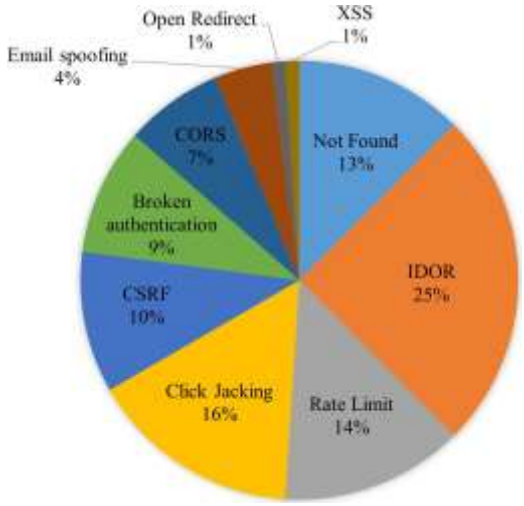

Fig. 4. **Percentage of vulnerability detection relative to sample size in web applications**
Source: **compiled by the [22, 23[, [24]**

This is a logical vulnerability in the project, through which hidden information can be obtained from the database using a permitted HTTP request. The essence of it is that identifiers that can be guessed are used in the request string. For example, if through the request "GET /product/25" you can get information about a product with identifier "25", then it is quite logical that you can substitute identifier "30", and the attacker will receive the corresponding product without interacting with the site.

Web application vulnerabilities arise as a result of several factors: the complexity of project implementation, poorly designed architecture, which requires adding more complex parts that were not even originally intended for implementation, and the human factor.

Regarding exploitation vulnerabilities – this is probably the most common cause of vulnerabilities

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

in web applications. It is a result that arises from implementation error vulnerabilities. They are the most systematized, and can be detected and eliminated at the stage of project development. These errors occur when attackers use vulnerabilities in software to perform malicious actions, such as leaking confidential information, making changes to the system, or harming users. Exploitation errors can be used to implement attacks such as cross-site scripting, SQL injection, use of dangerous functions, etc. Detecting and eliminating these errors at the development stage helps prevent potential attacks and ensure a high level of security of web applications. Additionally, exploitation errors may include taking advantage of weaknesses in web applications to carry out attacks such as injecting malicious code, stealing session files, exploiting insufficiently protected APIs, and other hacking techniques. Detecting and eliminating these errors requires thorough security auditing, following best practices in programming, and implementing security measures at all levels of web application development.

The essence of many client-side vulnerabilities is in attacks on web users' browsers. Such attacks are aimed at bypassing the “Same origin policy” (SOP). This is a protection mechanism that exists in any client browser. For example, when using CURL – there is no such protection. The purpose of such attacks is to use the “origin” from one resource in another. In fact, there is one mechanism that legitimately allows you to do this – cross-origin resource sharing (CORS). Therefore, to interact between two different client parts, you need to properly configure CORS security. Very often, developers neglect this security, and attackers often exploit vulnerabilities by executing JavaScript code.

Another common client-side vulnerability is related to using “Websockets”. Usually, to support such a connection, a request with “Connection: Upgrade” and other attributes is sent, primarily “origin”.

An example of a dangerous request can be depicted in Fig. 5.

```
GET /v1/subscribe/?xxxx HTTP/1.1
Connection: Upgrade
Upgrade: websocket
Origin: https://attacker.com
Sec-WebSocket-Version: 13
Cookie: session=123
Sec-WebSocket-Key: A/7665fKkoz9+CsKfHu7rQ==
```

Fig. 5. **WebSocket connection request example illustrating client-side vulnerability**
Source: **compiled by the authors**

The problem with this vulnerability is that the origin always needs to be validated. It must be valid, and the connection to the websocket should only be allowed if the origin is allowed by the server. This issue is relevant because even on Github there are many libraries for working with websockets. However, most of them do not have built-in protection and rely on the developer's responsibility. Accordingly, if the latter does not write any protection functionality – the application can be compromised through this vulnerability. Client-side vulnerabilities also include Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS), and others that annually occupy top positions in the OWASP Top Ten documentation. Let's take a closer look at the Insecure Design vulnerability.

2.3. Vulnerability analysis of Insecure Design

Insecure Design is a vulnerability that is inherent in the planning and architecture of a project, meaning it is considered within the framework of secure web application design. Unlike vulnerabilities like injections that can be avoided with a few lines of code, insecure design vulnerability can have unpredictable consequences for the entire web application, not just an individual component. The main cause of the vulnerability is improper threat modeling, which is done at the software design stage and propagates through to project completion. The vulnerability can arise during web application development when a developer disables important security features for testing and forgets to re-enable them. Another cause can be excessive information disclosure from HTTP request validation for the client or overly simple identity verification using easily guessable security questions.

One of the tasks facing the development team is project planning, which must take into account information security requirements on the one hand, as well as ensuring that all business processes and functionality do not conflict with each other, preventing “logical” errors from occurring. In this regard, Insecure Design vulnerability can arise due to an insufficient project budget, poorly thought-out business logic, and incorrect task setting for developers. Thus, a web application can be secure at a certain point, but after additional changes to software features, new opportunities open up for attacks on the web application.

A review of publications [25] shows that the Insecure Design vulnerability is still poorly formalized, and recommendations for protection may seem somewhat abstract.

The OWASP organization not only studies and defines lists of the most popular vulnerability categories, but also provides some recommendations for their elimination [26], in particular:

– use a secure development lifecycle and involve web application security professionals to evaluate and design security and privacy controls;

– create and utilize a library of secure design patterns;

– perform threat modeling against important business logic elements;

– implement checks at every level of the web application;

– full module and integration test coverage of the entire application;

– segmentation at the system and network layers according to exposure and protection needs;

– separation of duties across all tiers during design;

– resource usage limits per user or service.

Many researchers refer to this list, which is undoubtedly useful and developed based on many years of research and statistics. Most authors emphasize secure development (SDL) and involving web security experts. The question arises as to what measures companies and software developers can take if they do not have the resources to involve cybersecurity experts in their projects. In addition, for simple projects in terms of functionality, there is often no possibility of conducting additional cyber security expertise due to budget constraints. One way or another, the entire responsibility for project security lies with the development team, whose level of competence in cybersecurity may be insufficient to properly protect the web applications they create from "Insecure Design" vulnerabilities. This problem requires increased attention from professionals and the expansion and improvement of recommendations for web application developers.

In order to expand recommendations for protection against Insecure Design vulnerabilities at different software development stages, the secure development lifecycle (SDL) is further examined in this work.

2.4. Secure software development cycle

Secure Development Lifecycle (SDL) is a software development concept that involves clearly defined application requirements, secure coding, testing stages, certification, deployment, and sustainment [27, 28], [29, 30]. It is a process that enables maintaining the required level of system security during development and throughout its entire lifetime. The process flow is shown in Fig. 6.

Secure development lifecycle helps follow a systematic approach when creating secure web applications. The concept defines a set of actions and processes that should be applied throughout the entire development lifecycle – from formulating product requirements to taking it out of service.

Secure development lifecycle involves engaging security experts at all stages of system design and implementation. At the coding and testing stages, software compliance with security standards is verified and potential vulnerabilities are identified.

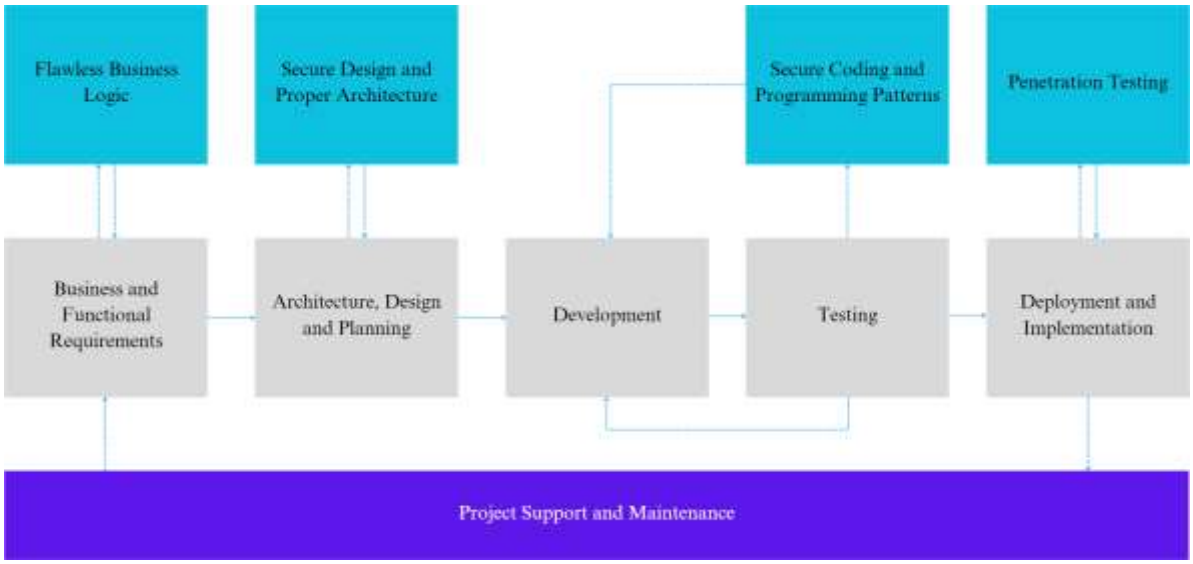

Fig. 6. **Secure development lifecycle**
Source: **compiled by the authors**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

An important component of SDL is the continuous monitoring of the deployed system to identify new threats, update protections, and fix vulnerabilities throughout the product's entire service lifetime. That is, Secure Development Lifecycle represents an integrated approach to security that covers the entire software lifecycle. This approach works well even when clients come with new ideas and ambitious additions or extensions of functionality, creating an environment where any client request is guaranteed to be analyzed from an information security perspective.

The choice of software architecture type does not have a decisive influence on the emergence of “Insecure Design” vulnerability, since before any code is written, developers must have a clear understanding of the architecture of the future web application, and therefore a plan to avoid or mitigate vulnerabilities.

2.5. Extending protection recommendations from Insecure Design

The Secure development lifecycle (SDL) is a process that enables the successful and secure development and maintenance of web applications. Since the emergence of SDL, it has been supplemented and improved with new recommendations, standardization, and successful practices based on the work of teams around the world. Since the “Insecure Design” vulnerability occupies the fourth position in the list of priority vulnerabilities, this necessitated a thorough examination of existing approaches and supplementing them with new ones to protect web applications at each stage of the SDL in this work.

The first important stages of the secure development lifecycle are defining business and functional requirements, and on their basis – planning the secure architecture of the project. When planning the architecture, it is important to consider future risks, the possibility of expanding the project, and the correctness of the approach to information exchange between components, especially with microservice architecture. Maintaining a balance between the functionality the client needs and what the development team can offer is an important caveat, since overloading with functionality can expose vulnerabilities to “Insecure Design”.

The development stage is a critical stage that comes after project planning. Often, developers who will be writing the code mistakenly think that everything has been planned out up to this point, all schemas and functional requirements are accounted for - so information security on the project is fine. This can lead to a situation where web application protections are not implemented or remain inadequate. In view of this, recommendations were developed, presented below in this section, regarding risk and threat analysis, and strengthening protection during the development stage.

The availability of disk space for regular users is essential. If we exclude cloud storage and consider simpler implementations, any website typically contains images, content files, and downloads. It's important to restrict access to the entire file system and allocate separate public logical spaces for user interaction with the file system. This way, vulnerabilities such as uncontrolled execution of uploaded files can be mitigated, and access modes (as mentioned in OWASP) can be limited.

Protection of requests for interacting with specific entities is crucial. First and foremost, never use clear and predictable identifiers of entity records in requests, as substituting them can grant access to other information in the project. Despite all warnings and examples of attacks, such a practice is not uncommon. In such cases, it's worth employing additional protection using various types of middleware. In simple terms, this is protection during a request to the server, which takes place between the start of the request and the execution of actions by a specific class. It's essential to verify the legitimacy of this request, whether the client can interact with the record related to the request, and whether such a record exists at all.

Validation of permissions to perform a certain request is crucial in software development. Often, developers make logical errors, trying to clearly indicate to the user what they are doing wrong at the moment, making it easier for them to correct their input and achieve the desired result. An example could be a detailed description of what was entered incorrectly during user authentication. Additionally, there is often a need to distinguish between permissions, and instead of issuing a 404 error, developers indicate a 403 error. However, for an attacker, this means that they have found an object to attack. Another example is WordPress CMS, where the use of its CSS classes with the "wp-" prefix and the standardized authentication endpoint “wp-login” make it easy for attackers to identify and exploit vulnerabilities.

Security when using APIs and libraries is crucial in software development. Developers often rely on pre-existing functionality to streamline their work, but they may overlook whether these components adhere to secure development lifecycle (SDL) practices, whether their code is updated, and

whether there is developer support. Carelessness in this matter can lead, among other things, to "supply chain attacks".

Standardization and programming patterns play a crucial role in software development. Utilizing programming patterns and principles such as SOLID ensures the writing of high-quality code. With correct programming approaches, issues like random code changes, leaving backdoors, console errors, and other unwanted consequences can be avoided. However, experimenting with new approaches, deviating from commonly accepted practices, and overcomplicating simple tasks can lead to problems associated with "Insecure Design".

The testing phase of a product is crucial in ensuring its quality. The quality of code coverage by tests heavily depends on the aforementioned programming patterns. The more intuitively simple the code is written, the easier it is to test it in the future. Testing the business logic, established at the beginning of the project, is essential. Often, developers "assure" that "everything works, it cannot be otherwise", and testing is done based on trust and the result of test coverage.

During the deployment phase, it is crucial to understand the entire journey of the web application, its essence, and functional requirements. Repositories or configuration files of the project after its completion may contain access, test, or sensitive data. Checking configuration files and repositories for confidential information will help avoid potential compromise of the web application or user data.

Supporting a project involves monitoring and continuously updating the software, as well as receiving feedback from users and analyzing the behavior of the web application under various operating conditions. Fig. 7 depicts an updated SDL process diagram, enhanced with recommendations and analysis provided earlier in this section.

Therefore, this addendum serves as an extension of recommendations for protection against vulnerabilities like "Insecure Design" within the context of the Secure Development Lifecycle (SDL) concept. It is noted that this concept is actively evolving worldwide with teams continuously adding new standards and best practices.

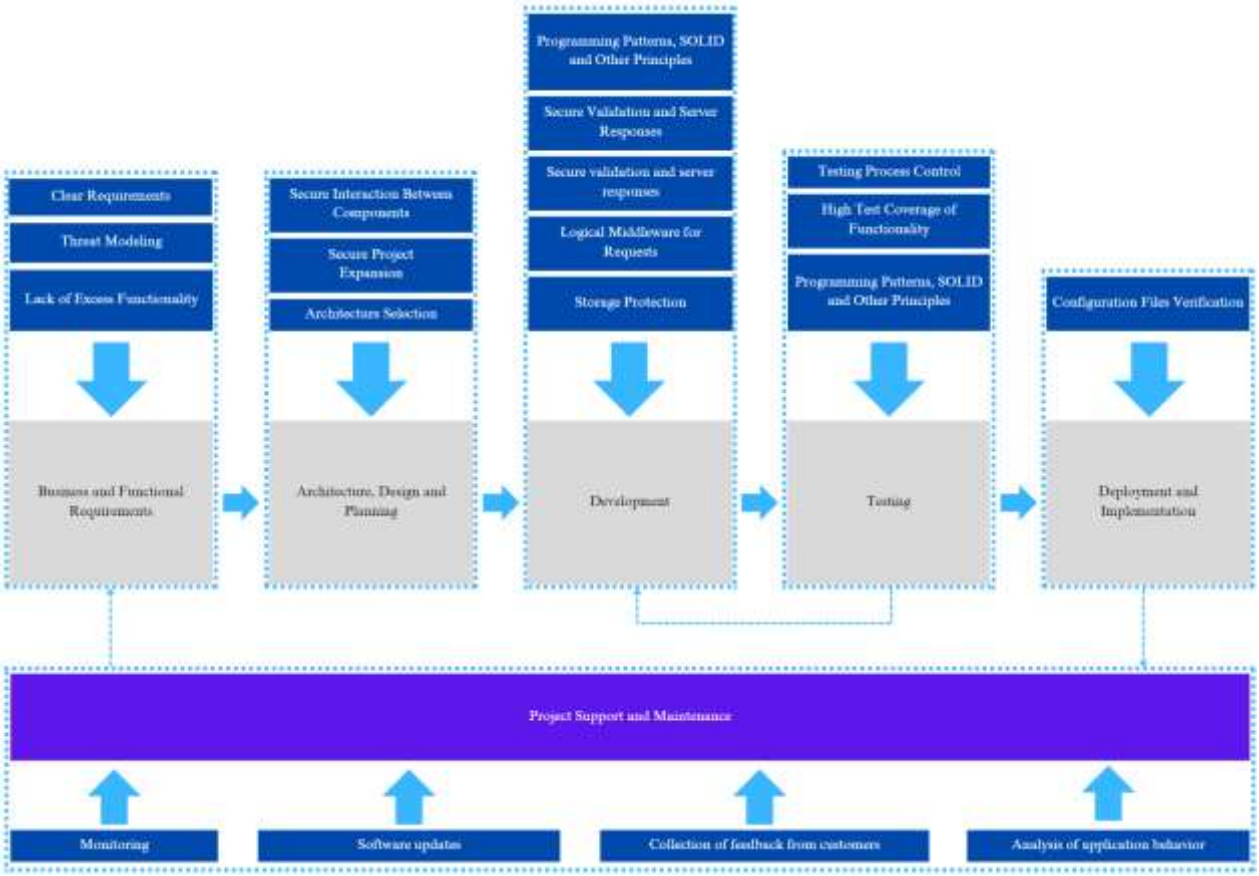

Fig. 7. **Addendum to secure development lifecycle**
Source: **compiled by the authors**

Special attention should be paid to the software development phase within the SDL framework. Therefore, a series of recommendations have been provided for developers to avoid common mistakes that could lead to security issues. This includes limiting access to the file system, protecting against request identifier substitution, securely working with APIs and libraries, adhering to coding standards, and more. Taking these recommendations into account will significantly enhance the security of developed web applications.

DISCUSSION OF RESULTS

The results of our research provide a deeper insight into the “Insecure Design” vulnerability issue within the context of web application development. By analyzing architectural vulnerabilities, we particularly focused on integrating security principles at the early stages of development. This approach helps to mitigate risks and ensures a strong foundation for creating safer web applications.

Further discussion of the results also highlights the importance of a comprehensive understanding of both technical and business aspects of the project. Security should not be considered in isolation from the main functionality and goals of the web application, as it is closely linked with all aspects of development.

An important aspect that requires further analysis is the impact of new technologies on the security of web applications. Technologies such as containerization, serverless architecture, and microservices open up new opportunities for optimization and scaling of web applications but also pose new challenges for ensuring security. In analyzing these technologies, it is important to understand not only their potential advantages but also the possible risks they may introduce to the architecture of web applications.

Considering these aspects in the context of our results shows that security measures need to be flexible and adapted to the rapidly changing digital landscape.

CONCLUSION

The article conducted an analysis of the "Insecure Design" vulnerability in web applications. It was established that protection against “Insecure Design” threats must be comprehensive and cover all stages of the Software Development Lifecycle (SDL).

It was found that the effectiveness of the proposed security measures can significantly depend on the specifics of the project and its architecture. This requires an adaptive approach to implementing security, where each web application requires an individual assessment of potential vulnerabilities and methods for their mitigation.

The proposed recommendations and diagrams can assist developers and security administrators in building protection against the "Insecure Design" vulnerability through modern methods and technologies, which open new possibilities for integrating security practices at various stages of the web application lifecycle. This not only enhances the level of protection for web applications but also provides flexibility in further management and support.

In conclusion, the importance of continuously updating and improving security practices cannot be overstated, as it is key to maintaining the reliability and security of web applications in the modern digital world.

REFERENCES

1. Lydon, M. “11 Companies who have recently faced a cyberattack”. – Available from: https://www.growbo.com/recent-cyber-attacks-on-companies. – [Accessed: Dec, 2023].

2. Kerner, S. M. “34 cybersecurity statistics to lose sleep over in 2023”. *TechTarget Network*. – Available from: https://www.techtarget.com/whatis/34-Cybersecurity-Statistics-to-Lose-Sleep-Over-in-2020. – [Accessed: Dec, 2023].

3. “2022 Sitelock annual website securite report”. *SiteLock A Sectige Company*. – Available from: https://s3.us-east-1.amazonaws.com/sectigo-sites-web/global/uploads/2022-SiteLock-Website-Security-Report-FINAL.pdf. – [Accessed: Dec, 2023].

4. Aydos, M., Aldan, C., Coşkun, E. & Soydan, A. “Security testing of web applications”. *A Systematic Mapping of the Literature, Journal of King Saud University – Computer and Information Sciences*, 2022; 34 (9): 6775–6792, https://www.scopus.com/authid/detail.uri?authorId=56928400400.
DOI: https://doi.org/10.1016/ j.jksuci.2021.09.018.

5. Upadhyay, D., Ware, N. & Mahesh, B. “Evolving trends in web application vulnerabilities: A comparative study of OWASP Top 10 2017 and OWASP Top 10 2021”. *International Journal of*

Engineering Technology and Management Sciences. 2023; 7 (6): 262–269. DOI: https://www.doi.org/10.46647/ijetms.2023.v07i06.038.

6. Fredj, O., Cheikhrouhou, O., Krichen, M., Hamam, H. & Derhab, A. "An OWASP Top TenDriven survey on web application protection methods". *Risks and Security of Internet andSystems*. 2021. p. 235–252. DOI: https://www.doi.org/10.1007/978-3-030-68887-5_14.

7. Trojanović, D. "Data of more than 200 million twitter users is leaked". – Available from: https://purplesec.us/security-insights/twitter-data-leak-200-million-users. – [Accessed: Dec, 2023].

8. Georgieva, E. "Linux malware targets 30+ WordPress Plugins". – Available from: https://purplesec.us/security-insights/wordpress-plugin-vulnerabilities. – [Accessed: Dec, 2023].

9. Abdulghaffar, K., Elmrabit, N., Yousefi, M. Enhancing "Web application security through automated penetration testing with multiple vulnerability scanners". *Computers.* 2023; 12 (11): 235, https://www.scopus.com/authid/detail.uri?authorId=16235087000.
DOI: https://doi.org/10.3390/computers12110235.

10. Mangal, L., Pushpendre. & Singh, P. "File Transferring web application using node JS". *International Journal for Modern Trends in Science and Technology*. 2022; 8 (1): 22–25. DOI: https://www.doi.org/10.46501/IJMTST0801004.

11. Poulter, A., Ossont, S. & Cox, S. "Using the MEAN stack to implement a RESTful service for an Internet of Things Application". *IEEE World Forum on Internet of Things*. 2015. p. 280-285. DOI: https://www.doi.org/10.1109/WF-IoT.2015.7389066.

12. Tilkov, S. & Vinoski, S. "Node.js: Using JavaScript to BuildHigh-Performance Network Programs". *IEEE Internet Com-puting*. 2010; 14 (6), 80–83. DOI: https://www.doi.org/10.1109/MIC.2010.145.

13. Wakil, K. "Extracting the features of modern web applications based on web engineering methods". *International Journal of Advanced Computer Science and Applications*. 2019; 10 (2). DOI: https://www.doi.org/10.14569/IJACSA.2019.0100209.

14. Wakil, K. & Jawawi, D. "Extensibility interaction flow modeling language metamodels to develop new web application concerns". *Kurdistan Journal of Applied Researc*h. 2017; 2 (3): 172–177. DOI: http://dx.doi.org/10.24017/science.2017.3.23.

15. Wakil, K. & Jawawi, D. "Model driven web engineering: A systematic mapping protocol" *Conference: ISCI.* – Available from: https://www.researchgate.net/publication/319839831_ Model_Driven_Web_Engineering_A_Systematic_Mapping_Protocol. – [Accessed: Feb, 2023].

16. "The PostgreSQL global development group. Chapter 55. Frontend/Backend Protocol. The PostgreSQL Global Development Group". – Available from: https://www.postgresql.org/docs/current/protocol.html. – [Accessed: Dec, 2023].

17. Faisal, N., Jianming, Y. & Xiaohui, T. "Classification of logical vulnerability based on group attacking method". *Procedia Computer Science*. 2020; 170: 923–928. DOI: https://doi.org/10.1016/j.procs.2020.03.109.

18. Sharma, C. & Jain, S. "Analysis and classification of SQL injection vulnerabilities and attacks on web applications". *International Conference on Advances in Engineering and Technology Research (ICAETR)*. 2014. p. 1–6. DOI: https://doi.org/10.1109/ICAETR.2014.7012815.

19. Li, X., Chang, X., Board, J. A. & Trivedi, K. S. "A novel approach for software vulnerability classification.". *Reliability and Maintainability Symposium (RAMS).* 2014. p. 1–7. DOI: https://doi.org/10.1109/RAM.2017.7889792.

20. Fournaris, A., PoceroFraile, L. & Koufopavlou, O. "Exploiting hardware vulnerabilities to attack embedded system devices: A survey of potent microarchitectural attacks." *Electronics.* 2017; 6 (3): 52. DOI: https://doi.org/10.3390/electronics6030052.

21. Yulianto, S., Abdullah, R. R. & Soewito, B. "Comprehensive analysis and remediation of insecure direct object references (IDOR) vulnerabilities in android APIs". *IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs).* 2023. p. 23–28. DOI: https://doi.org/10.1109/ICoCICs58778.2023.10276919.

22. Bhuiyan, T., Begum, A., Rahman, S. & Hadid, I. "API vulnerabilities: Current status and dependencies". *International Journal of Engineering & Technology*. 2018; 7 (2): 9–13. DOI: https://doi.org/10.14419/ijet.v7i2.3.9957.

23. Myers, B. & Stylos, J. "Improving API usability". *Communications of the ACM*, 2016; 59 (6): 62–69. DOI: https://doi.org/10.1145/2896587.

24. Scheller, T. & Kuehn, E. “Automated measurement of API usability: The API Concepts Framework”. *Information and Software Technology.* 2015; 61: 145–162. DOI: https://doi.org/10.1016/j.infsof.2015.01.009.

25. Yudin, O., Kharchenko, V. & Pevnev, V. “Scanning of web-applications: Algorithms and software for search of vulnerabilities “Code Injection” and “Insecure Design””. *IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS).* 2023. p. 1005-1010. DOI: https://doi.org/10.1109/IDAACS58523.2023.10348918.

26. “A04:2021 – Insecure Design. OWASP”. – Available from: https://owasp.org/Top10/A04_2021-Insecure_Design. – [Accessed: Dec, 2023].

27. Uçak, U. & Tuna, G. “Design, Development, and Testing of Web Applications: Security Aspects”. *Protecting User Privacy in Web Search Utilization, edited by Rafi Ullah Khan, IGI Global*, 2023. p. 117–138. DOI: https://doi.org/10.4018/978-1-6684-6914-9.ch006.

28. Gurung, G., Shah, R. & Jaiswal, D. “Software development life cycle models-a comparative study”. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*. 2020; 6 (4): 30–37. DOI: https://doi.org/10.32628/CSEIT206410.

29. Acharya, B. & Sahu, P. “Software development lifecycle models: a review paper”. *International Journal of Advanced Research in Engineering and Technology (IJARET)*. 2020; 11 (12). DOI: https://doi.org/10.34218/IJARET.11.12.2020.019.

30. Geogy, M. & Dharani, A. “Prominence of each phase in Software development life cycle contributes to the overall quality of a product”. *International Conference on Soft-Computing and Networks Security (ICSNS)*. 2015. DOI: https://doi.org/10.1109/ICSNS.2015.7292390.

Conflicts of Interest: The authors declare that there is no conflict of interest

Received 16.02.2024
Received after revision 29.04.2024
Accepted 14.05.2024

DOI: https://doi.org/10.15276/aait.07.2024.12
УДК 004.056

Удосконалення процесу SDL веб-додатків для запобігання вразливостям Insecure Design

Ревнюк Олександр Андрійович [1)]
ORCID: https://orcid.org/0009-0005-0511-5354; revo0708@gmail.com
Загородна Наталія Володимирівна [1)]
ORCID: https://orcid.org/0000-0002-1808-835X; zagorodna_n@tntu.edu.ua. Scopus Author ID: 57189380553:
Козак Руслан Орестович [1)]
ORCID: https://orcid.org/0000-0003-1323-0801; ruslank@tntu.edu.ua. Scopus Author ID: 57193443499
Карпінський Микола Петрович [2)]
ORCID: https://orcid.org/0000-0002-8846-332X; mikolaj.karpinski@up.krakow.pl. Scopus Author ID: 57202467671
Флуд Любомир Олегович [3)]
ORCID: https://orcid.org/0000-0002-8347-4265; flud@nltu.edu.ua. Scopus Author ID: 57202467671
[1)] Тернопільський національний технічний університет імені Івана Пулюя, вул. Руська, 56. Тернопіль, 46001, Україна
[2)] Університет Національної Освітньої Комісії, вул. Подхоражих, 2. Краків, 30-084, Польща
[3)] Український національний лісотехнічний університет, вул. Ген. Чупринки, 103. Львів, 79057, Україна

АНОТАЦІЯ

Згідно з останнім списком “OWASP Top Ten”, вразливість “Insecure Design” є одним з ключових факторів, що впливають на рівень захисту даних та функціональної надійності. Посилення уваги до цієї проблематики є актуальним, оскільки дана вразливість вперше з'явилася в списку OWASP і лише коротко описана в ньому. Дане дослідження спрямоване на виявлення та аналіз архітектурних вразливостей веб-додатків, що виникають внаслідок “Insecure Design”. Мета полягає не лише у виявленні конкретних вразливостей у процесі розробки та реалізації веб-додатків, але й у розробці детального переліку рекомендацій, які допоможуть не лише уникнути подібних проблем у майбутньому, але й створити

хорошу основу для безпечної розробки веб-додатків з самого початку. Для того, щоб побудувати системний підхід до безпеки на всіх етапах розробки, тут розглядаються рекомендації зі стандарту «Життєвий цикл розробки програмного забезпечення» (SDL). Особлива увага приділяється інтеграції принципів безпеки на всіх етапах життєвого циклу розробки. Аналіз базується на дослідженні існуючих архітектурних рішень, вивченні вразливостей та розробці методів їх усунення. Розроблений набір рекомендацій щодо підвищення безпеки веб-додатків включає заходи з архітектурного проектування, процесів верифікації та валідації, а також раннього виявлення потенційних вразливостей. Значну увагу приділено розробці безпечного коду, впровадженню політик безпеки та організації навчання розробників. Дослідження підкреслює важливість інтеграції безпеки в процес розробки веб-додатків з самого початку. Наукова новизна полягає в систематизації та розробці підходів до виявлення та усунення архітектурних вразливостей, спричинених “Insecure Design”. Практична значущість роботи полягає у підвищенні рівня безпеки веб-додатків, зниженні ризиків для бізнесу та користувачів, а також у формуванні культури безпеки серед розробників.

Ключові слова: Insecure design; веб-додатки; безпечний цикл розробки; практики безпеки; вразливості додатків; багаторівнева структура

ABOUT THE AUTHORS

Oleksandr Revniuk - graduate student of Cybersecurity Department. Ternopil Ivan Puluj National Technical University, 56, Ruska Str. Ternopil, 46001, Ukraine
ORCID: https://orcid.org/0009-0005-0511-5354; revo0708@gmail.com.
Research field: Cybersecurity; web application security assessment quality; web development; web application security

Ревнюк Олександр Андрійович - аспірант кафедри кібербезпеки. Тернопільський національний технічний університет імені Івана Пулюя, вул. Руська, 56. Тернопіль, 46001, Україна

Nataliya Zagorodna - Associate Professor, Head of Cybersecurity Department. Ternopil Ivan Puluj National Technical University, 56, Ruska Str. Ternopil, 46001, Ukraine
ORCID: https://orcid.org/0000-0002-1808-835X; zagorodna_n@tntu.edu.ua. Scopus Author ID: 57189380553.
Research field: Computer security and cryptography; data mining and analysis; game theory and decision science; signal processing; operations resear

Загородна Наталія Володимирівна – доцент, завідувачка кафедри Кібербезпеки. Тернопільський національний технічний університет імені Івана Пулюя, вул. Руська, 56. Тернопіль, 46001, Україна

Ruslan Kozak - Associate Professor, Cybersecurity Department. Ternopil Ivan Puluj National Technical University, 56, Ruska Str. Ternopil, 46001, Ukraine
ORCID: https://orcid.org/0000-0003-1323-0801; ruslank@tntu.edu.ua. Scopus Author ID: 57193443499.
Research field: Application security; security architecture; threat modeling

Козак Руслан Орестович – доцент, кафедра Кібербезпеки. Тернопільський національний технічний університет імені Івана Пулюя, вул. Руська, 56. Тернопіль, 46001, Україна

Mikolaj Karpinski - Professor of Cybersecurity Department University of the National Education Commission, 2, Podchorążych Str. Krakow, 30-084, Poland
ORCID: https://orcid.org/0000-0002-8846-332X; mikolaj.karpinski@up.krakow.pl. Scopus Author ID:57202467671
Research field: Cybersecurity; computer and sensor networks; Internet of Things; security of wireless networks; particularly in areas cryptography; cryptanalysis and DDoS attacks; lighting engineering; electric and photometric measurements

Карпінський Микола Петрович - професор кафедри Кібербезпеки. Університет Національної Освітньої Комісії, вул. Подхоражих, 2. Краків, 30-084, Польща

Liubomyr O. Flud - Associate Professor, Department of Information Systems and Computer Modeling. Ukrainian National Forestry University, 103, Gen. Chuprynky Str. Lviv, 79057, Ukraine
ORCID: https://orcid.org/0000-0002-8347-4265; flud@nltu.edu.ua. Scopus Author ID:57202467671
Research field: Mathematical and software for automation of modelling and calculation of physical and mechanical processes; mathematical modelling of wood deformability

Флуд Любомир Олегович - доцент кафедри Інформації системного та комп'ютерного моделювання. Український національний лісотехнічний університет, вул. Ген. Чупринки, 103. Львів, 79057, Україна